TITLE:

Ptolemaic Graph 上の最長路問題に関する研究(計算機科学の理論とその応用)

AUTHOR(S):

高原, 祥浩; 寺本, 幸生; 上原, 隆平

CITATION:

高原, 祥浩 ...[et al]. Ptolemaic Graph 上の最長路問題に関する研究(計算機科学の理論とその応用). 数理解析研究所講究録 2007, 1554: 109-116

ISSUE DATE:

2007-05

URL:

http://hdl.handle.net/2433/80965

RIGHT:

京都大学学術情報リポジトリ
KURENAI 紅
Kyoto University Research Information Repository

数理解析研究所講究録
第1554巻 2007年 109-116

Ptolemaic Graph上の最長路問題に関する研究

高原 祥浩, 寺本 幸生, 上原 隆平
Yoshihiro Takahara, Sachio Teramoto, Ryuhei Uehara
北陸先端科学技術大学院大学 情報科学研究科
Japan Advanced Institute of Science and Technology (JAIST)

概要

Ptolemaic Graphとは、Ptolemaic inequality を満たし、Chordal Graph と Distance Hereditary Graphの共通部分となるグラフクラスである。Ptolemaic GraphはClique Laminar Treeと呼ばれる木表現を持ち、この性質を利用してハミルトン閉路問題が線形時間で解けることが示されている。本論文ではこの結果を拡張し、Ptolemaic Graph上のハミルトン路問題・最長閉路問題・最長路問題を多項式時間で解くアルゴリズムを提案する。

1 研究の背景と目的

現在、いくつかのグラフクラスにおいては、ハミルトン(閉)路問題や最長(閉)路問題が多項式時間で計算可能かどうか証明されていない。特に最長路問題に関しては殆ど研究が行われておらず、効率的に解けるグラフクラスは限られている。

Ptolemaic Graphとは、Ptolemaic inequality を満たし、Chordal Graph と Distance Hereditary Graphの共通部分となるグラフクラスである。Ptolemaic Graph上でハミルトン(閉)路を求める線形時間アルゴリズムは存在するが[1, 7]、最長(閉)路問題については効率的な解法が知られていない。

Ptolemaic Graphに関する重要な特徴づけとして、上原・宇野による包含関係に基づいた木構造による表現が挙げられる[7]。いくつかの木表現をもつグラフクラスに対しては、動的計画法を使うことで最長路を多項式時間で見つけることが出来る。そのため、Clique Laminar Treeと呼ばれるこの木表現は、最長路問題解決の糸口となる。

上原・宇野により、木表現を利用した動的計画法でPtolemaic Graph上のハミルトン閉路問題が線形時間で解けることが示されている[7]。

本研究では、Ptolemaic Graphの木表現が持つ性質と、それを使ったハミルトン(閉)路問題の解法を拡張し、最長(閉)路問題を効率よく解くことを目的とする。

[1]の方法ではClique Laminar Treeと比較して弱いグラフ構造を用いており、最長(閉)路問題への拡張は困難であると考えられる。本研究では、まずClique Laminar Treeを用いたハミルトン閉路問題の解法を示し、ハミルトン路問題と最長(閉)路問題の解決に拡張する。

ハミルトン路問題は全ての頂点を接続する必要があるという点で、ハミルトン閉路問題と同質である。しかしパスの端点を計算する過程が必要となり、ハミルトン閉路問題よりも複雑である。

最長閉路は最長という条件のためハミルトン閉路とは異なり、どんなグラフ上にも存在する。全ての頂点を使用する必要が無く、頂点の取捨選択が要求される点がハミルトン問題と比較して困難である。

Ptolemaic Graphにおける最長路問題は、ハミルトン路問題と最長閉路問題の性質を併せ持つ問題であり、パスを求める手続きと、頂点を選別する手続きを組み合わせる必要がある。

2 Ptolemaic Graph

2.1 定義

Ptolemaic Graphとは、任意の4頂点 x,y,z,w に対して、Ptolemaic inequal-

ity $d(x,y)d(z,w) \leq d(x,z)d(y,w) + d(x,w)d(y,z)$ を満たすグラフであり、平面幾何的な性質をもつ。図 1 左は Ptolemaic Graph の例である。

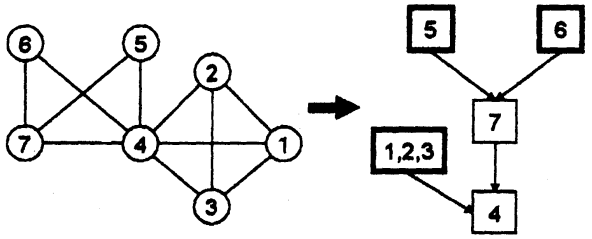

図 1: Ptolemaic Graph とその木表現の例

Ptolemaic Graph に関して、以下の定理が存在する。

定理 1. [5] 次の 3 つの命題は等しい：(1) G は Ptolemaic Graph; (2) G は distance hereditary[1] かつ chordal[2]; (3) 空でない共通部分を持つ任意の異なる極大クリーク P と Q に対して、$P \cap Q$ は $P \setminus Q$ と $Q \setminus P$ を分割する。

2.2 木表現

Ptolemaic Graph $G = (V, E)$ に対して、極大クリークの集合 $M(G)$ と、G の 2 つ以上の極大クリークの共通部分の集合 $L(G)$ を考える。このとき、$C(G) = M(G) \cup L(G)$ で定義される集合 $C(G)$ について以下の補題が存在する。

補題 1. [7] $G = (V, E)$ を Ptolemaic Graph とする。$C_1, C_2 \in C(G)$ が overlap[3] するとき、$C_1 \cap C_2$ は $C_1 \setminus C_2$ と $C_2 \setminus C_1$ を分割する。

Ptolemaic Graph $G = (V, E)$ は、集合 $C(G)$ を頂点集合とする、一意に定まる有向な木表現 $\vec{T}(C(G)) = (C(G), A(G))$ を持つ [7]。有向辺 (以下、弧) の集合 $A(G)$ は、各クリークの包含関係を表現しており、弧 (C_i, C_j) が $A(G)$ に含まれる必要十分条件は、$C_i \subset C_j$ かつ、$C_i \subset C \subset C_j$ を満たす C が存在しないことである。

[1]Distance Hereditary Graph とは、任意の誘導部分グラフがにおいて 2 頂点間の距離が変わらないグラフである。

[2]Chordal Graph とは、少なくとも長さが 4 である任意のサイクルが必ず弦を持つグラフである。

[3]集合 X と Y に対して、$X \cap Y \neq \phi, X \setminus Y \neq \phi$ そして $Y \setminus X \neq \phi$ が成立するとき、X と Y は overlap するという。

$L(G)$ の要素は G のセパレータとなる (補題 1)。またノード間の包含関係は層 (laminar) 構造をもつことから、この木表現は Clique Laminar Tree と呼ばれる。$\vec{T}(C(G))$ の有向辺を無向辺で置き換えたものを $T(C(G))$ と表記する。Clique Laminar Tree について、以下の定理が存在する。

定理 2. [7] $G = (V, E)$ が Ptolemaic であるのは、$T(C(G))$ が木であるとき、かつそのときに限る

与えられた Ptolemaic Graph に対し、Clique Laminar Tree は線形時間で構築できることが示されている。図 1 左は図右の Ptolemaic Graph を木表現に変換したものである。極大クリーク $M(G)$ が 2 重のノード、それらの共通部分 $L(G)$ が通常のノードに対応している。本稿では Ptolemaic Graph の構成要素である頂点 v に対し、Clique Laminar Tree 上の頂点集合 $C(G)$ に含まれる要素 (G 上のクリーク) $\mathcal{C}$ をノードと呼ぶ。また、冗長さを除いた木表現上で、ノード $\mathcal{C}$ が持つ頂点集合を $\ell(\mathcal{C})$ で表し、$\mathcal{C}$ の所持頂点と呼ぶ。このとき、$\mathcal{C}$ の子孫が持つ頂点は $\ell(\mathcal{C})$ に含まれない。

3 ハミルトン閉路問題

以下で、Clique Laminar Tree を用いた、Ptolemaic Graph 上のハミルトン閉路を求める [7] のアルゴリズムについて説明する。本稿ではこれを $\mathcal{HC}$ と呼ぶ。$\mathcal{HC}$ は、ハミルトン閉路問題を木表現における各ノードごとの部分問題に分割し、それぞれのノードで局所的なハミルトン閉路を計算する。以下でその計算方法について述べる

処理ノードを L、L の親を $P_1, P_2, ..., P_{p(L)}$、$i = 1, ..., p(L)$ とする。まず簡単のために、$c(L) = 0, p(L) \geq 0$ である場合について考える。

L が持つ頂点を使い、長さ $|L|$ のサイクルを一時的に L 内で形成する。このサイクルが持つ辺を、L の親 P_i ごとに 1 本ずつ割り当て、その辺を拡張して P_i の所持頂点を接続することにより、P_i をサイクルの一部とする (図 2)。補題 1 より、G から $L(G)$ の要素である L を取

り除くことによって、Gは$p(L)$個の連結成分に分解される。よって次の補題が成立する。

補題 2. [7] Ptolemaic Graph $G=(V,E)$ の Clique Laminar Tree $\vec{T}(C(G))$ が与えられたとき、$c(L)=0$ である $C(G)$ の要素 L に対して、以下のことが成立する。

(i) $|L| \geq p(L)$ のとき、Lは$p(L)$個の親を接続したサイクルを形成し、$|L|-p(L)$本の辺を余らせることが出来る。

(ii) $|L| < p(L)$ のとき、接続できない親があるため、Gはハミルトン閉路を持たない。

また、余った辺の数をmargin $m(L)$とする。つまり $m(L)=|\ell(L)|-p(L)$ である。

Clique Laminar Tree上のノードは層構造を持つため、余剰辺は先祖の接続に利用できる。ノード間の margin の配分は、distributionを用いて表現する。distributionは、それぞれの弧 $(C_j,C_i) \in \vec{T}(C(G))$ に対して割り当てられる関数 $\delta((C_j,C_i))$ として定義される。$i=1,...,p(L)$ に対して、それぞれの弧 (P_i,L) は、$\sum_{i=1}^{p(L)} \delta((P_i,L)) = m(L)$ となるようなdistribution $\delta((P_i,L))$ を持つ。各ノード間の親子関係は、クリークの包含関係を表したものであるため、余剰辺であるmarginは子から親に対してのみ配分可能になる。

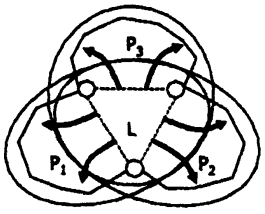

図 2: 辺の割り当て

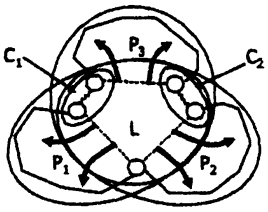

図 3: 親と子の接続

続いて、処理ノードLが子を持つとき、つまり $c(L)>0, p(L) \geq 0$ である場合を考える。Lの親・子をそれぞれ $P_1,P_2,...,P_{p(L)}$・$C_1,C_2,...,C_{c(L)}$ で表す。$i=1,...,c(L)$、$j=1,...,p(L)$ とすると、$C_i \subset L \subset P_j$ が成立する。このときのLにおける局所的なハミルトン閉路の構成法について示す。

それぞれの弧 (L,C_i) は、distribution $\delta((L,C_i))$ を持つ。$\delta((L,C_i))$は、C_iの処理時にC_iがLに提供可能な辺の余剰分であったが、Lの処理においては、LがC_iから提供される余剰辺の数を表す。以下の議論において、$m(C_i)$ と $\delta((L,C_i))$ は既に計算されているものとする。

Lの子C_iを、Lの所持頂点として利用することが可能である。まず、Lに対して割り当てられているC_iの辺を1本切り取り、Lとその親の頂点を使ったサイクルの一部として置き換える(図3)。もし、Lの子のあるC_iに対して $\delta((L,C_i))>0$ ならば、P_jを接続するための頂点としてC_iを使うことが出来る。よってLが子を持つ場合のmargin $m(L)$は、$m(L)=|\ell(L)|+c(L)+\sum_{i=1}^{c(L)}\delta((L,C_i))-p(L)$ と定義される。以上の議論から次の定理が導かれる。

定理 3. [7] $G=(V,E)$ を Ptolemaic Graph とする。Gがハミルトン閉路を持つ必要十分条件は、$C(G)$におけるそれぞれのノードLについて、$m(L) \geq 0$ を満たすようなmarginのdistributionが存在することである。

$\mathcal{HC}$は未処理の隣接ノードが1つになったノードに対して、局所的なハミルトン路の形成を逐次行う。このとき同じノードを複数回処理する必要がないことが、次の補題からいえる。

補題 3. Ptolemaic Graph $G=(V,E)$ のハミルトン閉路を $(x_0,x_1,x_2,...,x_k,x_0)$ とする。$T(C(G))$上の弧を持つ任意の異なるノードC_1、C_2を考えたとき、Gのハミルトン閉路 $(x_0,x_1,x_2,...,x_k,x_0)$は、$\ell(C_1)$に含まれる頂点と$\ell(C_2)$に含まれる頂点を接続する辺を高々2本しか持たないとしてよい。

証明. 略 □

補題3は、本稿で取り扱う他の経路問題に関しても同様に成立する。

処理ノードLに対して、唯一の未処理であるLの隣接ノードを、次の処理ノードXとする。Xを持つノードから計算するため、$\mathcal{HC}$は葉から処理を行う。Xが親であれば、余剰頂点の提供は可能だが、不足分の要求は出来ない。よって、$\delta((X,L)) \geq 0$ ならその余剰分をXに伝達し、$\delta((X,L))<0$ ならハミルトン閉路は無いと判断する。Xが子の場合は不足頂点の要求は可能だが、余りを提供できない。そのため、$\delta((X,L)) \geq 0$ となり余りが発生した場合

は、$\delta((L,X)) = 0$ とし、不足分は $\delta((L,X))$ に正の値として格納して X に伝達する。

以上の議論から、アルゴリズム $\mathcal{HC}$ に対して次の補題が成り立つ。

補題 4. [7] Ptolemaic Graph $G=(V,E)$ が与えられ、$\mathcal{HC}$ を実行したとする。このとき G がハミルトン閉路を持つ必要条件は、$T(C(G))$ 上の任意のノード L と、次に処理したノード X に対して、$\delta \geq 0$ が成立することである。

$G=(V,E)$ とし、$|V|=n$、$|E|=m$ とする。このとき $\vec{T}(C(G))$ が持つノードの数 $|C(G)|$ を n' と置くと $n'=O(n)$ となり、アルゴリズム $\mathcal{HC}$ は計算時間、領域ともに $O(n')=O(n)$ である [7]。

4 ハミルトン路問題

サイクルではなくパスを求めるため、どの2頂点をパスの端点とするか判断するプロセスが必要になる。そのためハミルトン閉路問題よりも難しい。

4.1 概要

提案アルゴリズムを $\mathcal{HP}$ とする。$\mathcal{HP}$ は Clique Laminar Tree 上のノードの組 (S,T) を端点を含むクリークとしたハミルトン路の有無を検査する。

固定した2頂点間の処理であること、サイクルではなくパスを求めることを除けば、基本的な手続きは $\mathcal{HC}$ とほぼ同じであり、ハミルトン閉路問題で定義された margin の distribution を利用して解く。

4.2 動的計画法に基づくアルゴリズム

$\mathcal{HP}$ は、はじめに Clique Laminar Tree 上で2つのノード (S,T) を固定する。このとき、(S,T) を端点とする無向のパス $(S,...,T)$ が決まる。このパス上にあるノードを除く全てのノードに対して、事前処理として margin の distribution を計算しておく。このとき、margin の不足を要求出来ないノードが存在すれば、S,T を端点を含むクリークとするハミルトン路は存在せず、他の2頂点の組み合わせを考える必要がある。事前処理の後、計算済みの distribution を利用し、$(S,...,T)$ 上の各ノードで局所的なハミルトン路の計算を繰り返す。L の直前に処理したノード W と次に処理するノード X が親か子かによって、局所的なハミルトン路の構成方法が変わる。

$(S,...,T)$ 上の全てのノードが処理済となり、ハミルトン路があればそれを出力する。そうでなければ、$C(G)$ に含まれる他の2つのノードの組み合わせに対して同様の処理を行う。$S=T$ のときは、ハミルトン閉路を求めることになるため、以降では $S \neq T$ であるとする。

固定した (S,T) 間の各ノードにおける、局所的なハミルトン路の計算について、サイクルの形成とは異なる部分がある。処理ノード L において、k 個の先祖をサイクルの一部として接続するために必要な辺の数は k 本である。しかし、L 内の頂点を使って一時的にパスを作ると、それだけでは頂点数 -1 本の辺しか得られない。さらに、このパスの両端点から辺が伸ばせるかどうかという条件も、L の直前に処理したノード W と次に処理するノード X が、それぞれ親か子かによって変化する。これらのことは margin と distribution の計算に影響を与える。詳細な説明は省略する。

Ptolemaic Graph $G=(V,E)$ が与えられ、$T(C(G))$ 上のノードの組 (S,T) について $\mathcal{HP}$ を実行したとする。このとき、以下の補題が成立する。

補題 5. G が S,T 内に端点を含むハミルトン路を持つ必要条件は、$(S,...,T)$ 上で最初に処理したノード S と、次に処理したノード X に対して、X が親のとき $\delta((S,L)) \geq -1$ が成立することである。

補題 6. G が S,T 内に端点を含むハミルトン路を持つ必要条件は、$(S,...,T)$ 上の S,T を除く任意のノード L と、L の直前に処理したノード W、次に処理したノード X に対して、以下の条件が成立することである。

(i) W が子かつ X が親のとき、$\delta((X,L)) \geq 0$ である。

(ii) W と X が親のとき、$\delta((X,L)) \geq -1$ である。

補題 7. G が S,T 内にそれぞれ端点を含むハミルトン路を持つ必要十分条件は、$(S,...,T)$ 上で最後に処理したノード T に対して、$m(T) \geq 0$ が成り立つことである。

いずれの場合の処理も、1つのノードにかかる処理時間は定数時間内に抑えられることは自明である。(S,T) の組み合わせは ${}_nC_2$ 通りあり、次の定理が導かれる。

定理 4. $G=(V,E)$ を Ptolemaic Graph としたとき、$\mathcal{HP}$ は G 上のハミルトン路の有無を $O(n^3)$ 時間、$O(n)$ 領域で判定する。

5 最長閉路問題

すべての頂点を使うという条件がないため、使わない頂点を選ぶプロセスが必要となり、ハミルトン閉路問題よりも複雑になる。

5.1 概要

Ptolemaic Graph 上で最長閉路を求めるアルゴリズム $\mathcal{LC}$ を提案する。$\mathcal{LC}$ は動的計画法に基づき、Clique Laminar Tree 上の各ノードについて局所的な最長閉路の候補を計算する。

処理ノードを L、次の処理ノードを X とする。L の処理済の隣接ノードは全て、これまで計算してきた局所的な最長閉路の候補を持つ。処理済の隣接ノードが持つこの情報は、利得表と呼ばれる配列によって管理される。利得とは接続可能なパスの長さ表す。以下で利得表について説明する。

5.2 利得表

処理ノード L の親を $P_1,P_2,...,P_{p(L)}$、子を $C_1,C_2,...,C_{c(L)}$ とし、$i=1,...,p(L)$、$j=1,...,c(L)$ とする。

- 処理ノードの親が持つ利得表について
  L の親 P_i が持つ利得表を δ_{P_i} とする。δ_{P_i} は、処理ノード L が P_i に k 本の辺を提供したときに得られる利得 $\delta_{P_i}[k]$ を管理する配列である。$\delta_{P_i}[k]$ には、先祖のノードの中で、所持頂点数が大きな順に k 個のノードの所持頂点の和が格納されている。
- 処理ノードの子が持つ利得表について
  L の子 C_i が持つ利得表を γ_{C_j} とする。γ_{C_j} は、処理ノード L が C_i に k 本の辺を要求したときに付随する利得 $\gamma_{C_j}[k]$ を管理する配列である。$\gamma_{C_j}[k]$ には、子孫から得られる最大利得から、所持頂点の少ない順に k 個のノードの所持頂点の和を差し引いたものが格納されている。

それぞれの利得表が持つ要素の数 (サイズ) を、$|\delta_{P_i}|$、$|\gamma_{C_j}|$ と表す。利得表について、以下の補題が成立する。

補題 8. 利得表 $\delta_{P_i},\gamma_{C_j}$ に対して、以下のことが成り立つ。

1 $k<k'$ のとき、$\delta_{P_i}[k] \leq \delta_{P_i}[k']$ が成り立つ。

2 $1<k<|\delta_{P_i}|$ である k に対して、$(\delta_{P_i}[k]-\delta_{P_i}[k-1]) \geq (\delta_{P_i}[k+1]-\delta_{P_i}[k])$ が成り立つ。

3 $k<k'$ のとき、$\gamma_{C_j}[k] \geq \gamma_{C_j}[k']$ が成り立つ。

4 $1<k<|\gamma_{C_j}|$ である k に対して、$(\gamma_{C_j}[k-1]-\gamma_{C_j}[k]) \leq (\gamma_{C_j}[k]-\gamma_{C_j}[k+1])$ が成り立つ。

証明. 略 □

また、実際の処理では処理ノード L に対して、処理済の親、子が持つ利得表を全て統合し、それぞれを δ、γ とする。統合前と後で利得表が持つ意味は変化しない。

補題 9. 利得表 δ_{P_i} と γ_{C_j} は $O(n\log n)$ 時間、$O(n)$ 領域で統合可能であり、統合後の利得表 δ、γ が示す意味は変化しない。

証明. 略 □

5.3 動的計画法に基づくアルゴリズム

$\vec{T}(C(G))$ 上のノードに対する具体的な処理方法を以下に示す。この局所的な計算は未処理の隣接点が一つになったノードに対して行われる。処理ノード L において、次の処理ノード X が親の場合は δ_L を、子の場合は γ_L を計算する。また L の状態により、場合わけを行う。

(1) 葉の場合

$\mathcal{LC}$ は、はじめに Clique Laminar Tree が持つ

葉を全て処理する。この場合、L の所持頂点数 $|L|$ を、子 X に伝達する利得表 δ_L に格納する。

(2) 葉では無い場合
葉の処理後、$\mathcal{LC}$ は未処理の隣接ノードが1つになったノードに対して順次処理を行う。今後の議論のいずれの場合においても、はじめに親の利得表 δ_{P_i} と子の利得表 γ_{C_j} をそれぞれ統合し、δ と γ を得る。

(2-1) $p(L) > 0, c(L) = 0$ の場合
このとき、L に隣接する唯一の未処理ノード X は親となる。$\mathcal{LC}$ は δ と $\ell(L)$ を用いて、X に伝達する利得表 γ_L を構成する。L は子を持たないため、先祖の接続に使用できる辺の数は $|\ell(L)| = |L|$ 本である。

X に $k(0 \leq k \leq |L|)$ 本渡した場合、その他の先祖には残りの $|L| - k$ 本を提供できる。このとき L は、X を除く親から $\delta[|L| - k]$ 本の辺を受け取って、長さ $|L| + \delta[|L| - k]$ のサイクルを形成し、これを X に渡すことができる。よって $\gamma_L[k] = |L| + \delta[|L| - k]$ とする。ただし、$\gamma_L[0]$ は X に全く頂点を提供しない場合であり、L の処理時点で完結した局所的な最長閉路となるため、$\gamma_L[0]$ を除いた利得表を X に伝達する。$\gamma_L[0]$ は完結した最長閉路の候補を格納する配列 LC に別個格納する。

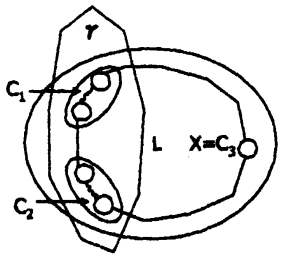

図 4: $p(L) > 0$ かつ $c(L) = 0$ の場合

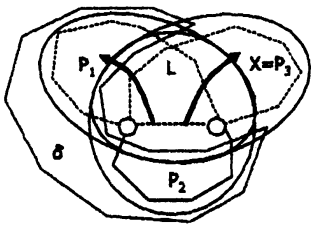

図 5: $p(L) = 0$ かつ $c(L) > 0$ の場合

(2-2) $p(L) = 0, c(L) > 0$ の場合
このとき、L は葉ではないルートであり、未処理である唯一の隣接ノード X は子となる。L は親を持たないため、γ から少なくとも1本の辺を拡張出来れば、X を含む L 内の頂点を全て接続可能である (図5)。表中で最大利得を持つ $\gamma[c(L)]$ を選択した場合でも、少なくとも処理済の子の数だけ辺が自由に使えるので、この場合のみを考える。よって、$\gamma[c(L)]$ と $|L|$ の和を δ_L に格納する。つまり $\delta_L[1] = |L| + \gamma[c(L)]$ となる。

(2-3) $p(L) > 0, c(L) > 0$ の場合
統合された利得表 δ、γ、そして $\ell(L)$ を用いて X への利得表を構成する。$p(L) > 0$ であるため、L 内で利用できる辺を先祖に振り分ける必要がある。また、$c(L) > 0$ であることから、L において利用可能な辺の数は子孫から貰った辺数によって決まる。そのため、はじめに子孫全体から $h(0 \leq h \leq |\gamma|)$ 個の頂点を受け取った場合の、先祖の接続に使用可能な頂点を表す $S_h(L)$ を計算する。以下で、このときに X が親である場合と子である場合の処理方法について、それぞれ示す。

(2-3-1) X が親の場合
S_h と利得表 δ を用いて、X への利得表 γ_L を作成する。このとき、L で使用可能な辺数は $S_h(L) = |L| + c(L) + h$ で定義される。h を固定し、$S_h(L)$ から k 本の辺を X に提供した場合について、利得表 γ_L に格納する値を計算していく。

L 内で使用可能な辺数を $S_h(L)$ として固定するため、その後の基本的な処理の流れは $p(L) = 0$ かつ $c(L) > 0$ の場合と同様である。X に k 本の辺を渡したとき、その他の先祖 (つまり δ) には $S_h(L) - k$ 本の辺が提供可能であり、$\gamma_L[k] = |L| + \gamma[h] + \delta[S_h(L) - k]$ とする。ただし、$(S_h(L) - k) > |\delta|$ の場合、δ に提供する辺の数が接続可能な先祖のノードの数を超えているため計算する必要がない。

この手続きは h を固定したものであるため、同様の処理を全ての h に対して行う。X に k 個渡したときの利得は、既に $\gamma_L[k]$ に格納されている値よりも大きければ更新する。$\gamma_L[0]$ の要素があればそれを配列 LC に保存する。

(2-3-2) X が子の場合
X に渡す利得表 δ_L を構成する。このとき X が L の子であるため、L で使用可能な辺数は $S_h(L) = |L| + c(L) + h + 1$ と表される。X が親の場合と同様、先祖をつなぐ為に $S_h(L)$ から何個の頂点を使うのか、全ての h について考える。

X 自体は L の子であるため、$S_h(L)$ は全て δ に配分すればよい。よって、$\delta_L[1] = |L| + \gamma[h] + \delta[S_h(L)]$ となる。このとき、$\delta[S_h(L)]$ 本の辺が先祖から得られるが、$|\delta| > S_h(L)$ であ

るときは接続できていない先祖が存在する。そのため、δ_L に未使用の δ の要素を加えていき δ_L を完成させる。つまり $k>1$ に対し、$\delta_L[k] = \delta_L[k-1]+\delta[S_h(L)+k-1]-\delta[S_h(L)+k-2]$ となる。

全ての h について、上記の利得表 δ_L の計算を行う。ある k について計算した利得表 $\delta_L[k]$ は、既に格納されている値より大きければ更新する。

(3) 最後のノードの場合

L が最後の処理ノードであるとき、γ から何個の頂点を使い、L の所持頂点とあわせて δ から何個の頂点を獲得したか、全ての組み合わせを列挙し、配列 LC に格納する。配列 LC の要素の中で最大のものが、G の最長閉路の長さとして得られる。

以上の議論により、次の補題が成立する。

補題 10. Ptolemaic Graph $G=(V,E)$ の最長閉路として $(x_0,x_1,x_2,...,x_n,x_0)$ $(0\leq i,j\leq n)$ を考える。$\mathcal{LC}$ を実行したとき、$T(C(G))$ 上のノード L と、次に処理されたノード X に対して、$x_i\in L$ かつ $x_j\in X$ であるとする。このとき、以下のことが成立する。

- L が $p(L)=0$ かつ $c(L)>0$、$p(L)>0$ かつ $c(L)>0$ かつ X が子であるとき、L は利得表 δ_L を持ち、$\delta_L[k]$ は、X が L に k 本の辺を提供したときに得られる利得を表す。
- L が $p(L)>0$ かつ $c(L)=0$、$p(L)>0$ かつ $c(L)>0$ かつ X が親であるとき、L は利得表 γ_L を持ち、$\gamma_L[k]$ は、X が L に k 本の辺を受け取ったときに付随する利得を表す。

証明. 略 □

定理 5. $G=(V,E)$ を Ptolemaic Graph としたとき、$\mathcal{LC}$ は G の最長閉路の長さを $O(n^3)$ 時間、$O(n^2)$ 領域で計算する。

略証. 補題 8、9、10 より、各ノードの処理の際、親と子の利得表から最大利得を連結することで、処理ノードが持つ利得表を $O(n)$ 時間で構築 (更新) 出来る。$p(L)>0$ かつ $c(L)>0$ の場合は、h の取り方が高々 n 通りあり、その数だけ利得表の計算を行うため、1 つのノードの処理時間は $O(n^2)$ である。よって、全体の計算時間は $O(n^3)$ となる。また、利得表のサイズは高々 n であり、これを全ノードが持つとすると計算領域は $O(n^2)$ となる。 □

$\mathcal{LC}$ が出力するのは最長閉路の長さであるが、実際の最長閉路の出力も可能である。

6 最長路問題

Ptolemaic Graph における最長路問題は、前述したハミルトン路問題と最長閉路問題を組み合わせた問題であり、パスの端点を求める手続きと、頂点を選別する手続きが必要になる。従って、この 2 つの問題を解くアルゴリズムを組み合わせることで解決可能になる。

6.1 概要

提案アルゴリズムを $\mathcal{LP}$ とする。$\mathcal{LP}$ は $\mathcal{HP}$ と同様に、$\vec{T}(C(G))$ 上の固定した 2 つのノード (S,T) を端点を含むクリークとした最長路を求める。この作業を全ての頂点の組み合わせに対して行い、最も多くの頂点を使用したパスを G の最長路として出力する。$S=T$ のときは、最長閉路を求めることになるため、以降では $S\neq T$ であるとする。

$\mathcal{LP}$ は前処理として $\mathcal{LC}$ を利用し、$(S,...,T)$ 上のノード以外のノードに対して、利得表 δ と γ(局所的な最長閉路の候補を格納した配列) を計算しておき、$(S,...,T)$ 上のノードに伝達する。$(S,...,T)$ 上の処理ノード L に対する手続きは、直前の処理ノード W と次の処理ノード X の状態によって場合分けされる。それぞれの場合において、$\mathcal{LC}$ を利用した処理を行う。

$\mathcal{LP}$ が求めるのは $(S,...,T)$ 上のノードを使用した最長路である。よって、直前の処理ノード W と、次の処理ノード X に対しては、L は少なくとも辺を 1 本ずつ提供、または受け取り、(S,T) 間のパスが途切れないようにする必要がある。

また、$\mathcal{LC}$ はサイクルを求める手続きであるため、これをパスの形成に利用できるようにする必要がある。そのため、L 内で利用可能な辺の数の調整をそれぞれの L について行う。

6.2 動的計画法に基づくアルゴリズム

基本的には $\mathcal{HP}$ と $\mathcal{LC}$ の組み合わせとなるアルゴリズムのため、処理ノード L で利用可能な辺の数の調整について示した補題のみを記載し、詳細は省略する。

補題 11. Ptolemaic Graph $G=(V,E)$ の最長路として $(x_0,x_1,x_2,...,x_n)$ $(0\leq h,i,j\leq n)$ を考える。ある (S,T) を固定してLPを実行したとき、$T(C(G))$ 上のノード S、T、L、L の直前に処理されたノード W、次に処理されたノード X に対して、$x_0\in S$、$x_n\in T$、$x_h\in W$、$x_i\in L$ かつ $x_j\in X$ であるとする。このとき、以下のことが成立する。

(i) L に対して、L が S かつ X が親、L が T かつ W が親、W,X が親であるとき、L 内で利用可能な頂点数 $+1$ 本の辺が先祖の接続に利用可能。

(ii) L に対して、L が S かつ X が子、L が T かつ W が子、W が子かつ X が親、W が親かつ X が子であるとき、L 内で利用可能な頂点数分の辺が先祖の接続に利用可能。

(iii) L に対して、W,X が子であるとき、L 内で利用可能な頂点数 -1 本の辺が先祖の接続に利用可能。

$\mathcal{LP}$ は $\mathcal{HP}$ と $\mathcal{LC}$ を組みあわせたアルゴリズムであり、$O(n^3)$ 時間かかる $\mathcal{LC}$ の処理を、$\vec{T}(C(G))$ 上の全てのノードの組に対して行う。よって次の定理が成立する。

定理 6. $G=(V,E)$ を Ptolemaic Graph としたとき、$\mathcal{LC}$ は G の最長路の長さを出力し、計算時間は $O(n^5)$、計算領域は $O(n^2)$ である。

$\mathcal{LP}$ が出力するのは最長路の長さであるが、実際の最長路の出力も可能である。

7 まとめ

本稿では、木表現を用いてPtolemaic Graph上のハミルトン路、最長(閉)路問題を解く多項式時間アルゴリズムを提案した。今後の課題は各アルゴリズムの計算時間・領域の改善と、Distance Hereditary Graph上の最長(閉)路問題の解決が挙げられる。

参考文献

[1] Maw Shang Chang, Shaur Ching Wu, Gerard J. Chang and Hong Gwa Yeh, Hamiltonian Problems on Ptolemaic Graphs, Proceedings of Workshop on Algorithms and Theory of Computation, International Computer Symposium, Chiayi, Taiwan, R.O.C., pp.27–34, 2000.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, Introduction To Algorithms Second Edition, The MIT Press, 2001.

[3] Martin Charles Golumbic, Algorithmic Graph Theory and Perfect Graphs Second Edition, Elsevier B.V., 2004.

[4] Reinhard Diestel, Graph Theory, Springer-Verlag, 1997.

[5] Edward Howorka, A Characterization of Ptolemaic Graphs, Journal ofGraphs Theory, 5:323–331, 1981.

[6] Ryuhei Uehara and Yushi Uno, Efficient Algorithms for the Longest Path Problem, 15th Annual International Symposium on Algorithms and Computation (ISAAC 2004), Lecture Notes in Computer Science Vol. 3341, pp.871–883, 2004.

[7] Ryuhei Uehara and Yushi Uno, Laminar Structure of Ptolemaic Graphs and Its Applications, 16th Annual International Symposium on Algorithms and Computation (ISAAC 2005), Lecture Notes in Computer Science Vol. 3827, pp.186–195, 2005.